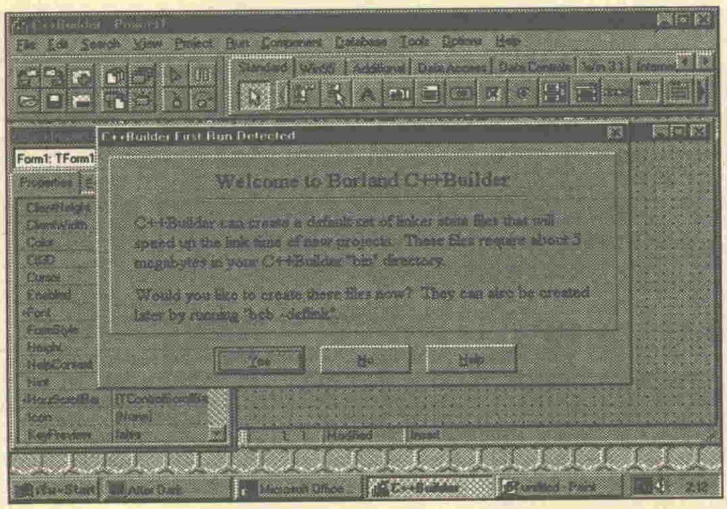
ธาวินทร์ ลีลาคุณารักษ์

SOFT MAIL

Borland C++ Builder



พบกับ Borland C++ Builder ที่สร้างเพื่อสนองตอบความต้องการของโปรแกรมเมอร์ C/C++ โดยเฉพาะ

ตั้งแต่บอร์แลนด์ผลิต Delphi เวอร์ชันแรก ซึ่งเป็นคอมพายเลอร์ภาษา Pascal ออกมาวางตลาดเมื่อ 2-3 ปีก่อนนี้ ทำให้ผู้ที่ถนัดภาษา C หรือ C++ โดยเฉพาะกลุ่มที่ใช้คอมพายเลอร์ตระกูล Turbo ของบอร์แลนด์เอง มีความคิดหวังว่าบอร์แลนด์คงจะผลิตคอมพายเลอร์ที่มีสภาพแวดล้อมแบบเดียวกับ Delphi ออกมาในไม่ช้านี้ และในที่สุดความหวังของโปรแกรมเมอร์ที่ใช้ภาษา C/C++ ก็เป็นจริงครับ เมื่อทางบอร์แลนด์ได้สร้าง Borland C++ Builder ออกวางจำหน่าย โดยรายละเอียดของคอมพายเลอร์ชุดนี้มีดังต่อไปนี้ครับ

Borland C++ Builder

Borland C++ Builder เป็นคอมพายเลอร์ภาษา C++ ที่ใช้สภาพแวดล้อมแบบ Visual Envelonment แบบเดียวกับ Delphi โดย Borland C++ Builder จะเป็นคอมพายเลอร์ที่ใช้สร้างแอพพลิเคชันแบบ 32 บิต ที่ทำงานบนวินโดวส์ เอ็นที หรือวินโดวส์ 95 ขึ้นไปเท่านั้น ไม่สามารถนำไปใช้สร้างแอพพลิเคชันแบบ 16 บิต ที่ทำงานบนวินโดวส์ 3.xx หรือบนดอสได้

สำหรับ Borland C++ Builder ที่เราพบเห็นในท้องตลาดตอนนี้ จะเป็นรุ่นที่ใช้สภาพแวดล้อมแบบเดียวกับ Delphi เวอร์ชัน 2.01 ซึ่งถ้าเราจะเปรียบเทียบส่วน Desktop ของโปรแกรม Borland C++ Builder กับโปรแกรมต่าง ๆ ที่ผลิตออกมาในปี 2540 (1997) นี้ Desktop ของ Borland C++ Builder ดูจะค่อนข้างล้าสมัยไปหน่อยครับ

เมื่อเราเข้าไปทำงานใน Borland C++ Builder เราจะพบว่าวิธีการกำหนดรายละเอียดของสภาพแวดล้อมต่าง ๆ (IDE) ก่อนเริ่มใช้ Borland C++ Builder สร้างโปรแกรมใช้งานจะคล้ายกับการจัดสภาพแวดล้อมของ Delphi มาก มีแตกต่างกันบ้างในบางจุด

และเมื่อเราเริ่มสร้างโปรแกรมประยุกต์ด้วยคอมพายเลอร์ชุดนี้ เราจะพบว่าวิธีคิดวิธีทำงานต่าง ๆ ของขั้นตอนของการสร้างเป็นตัวโปรแกรมที่เราจะใช้งานจะเหมือนกับการสร้างโปรแกรมบน Delphi ทุกประการ ทั้งนี้เนื่องจากซอร์สโค้ดที่ใช้สร้างเป็น Borland C++ Builder เป็นซอร์สโค้ดชุดเดียวกับที่ใช้สร้าง Delphi เวอร์ชัน 2.01 เพียงแต่มีการเพิ่มส่วนที่ช่วยเจนเนอเรจชุดคำสั่งเป็นภาษา C/C++ และเปลี่ยนแปลงกระบวนการคอมพายล์งานใหม่เท่านั้น ซึ่งในระหว่างที่เราใช้ Borland C++ Builder สร้างโปรแกรมใช้งาน เราจะพบว่ามีองค์ประกอบของโปรแกรมในบางส่วนยังคงมีชุดคำสั่งเป็นภาษา Pascal อยู่

Component ยกมาจาก Delphi ทั้งหมด

สำหรับสิ่งที่จะทำให้เราเห็นความแตกต่างระหว่าง C++ Builder กับ Delphi ชัดเจนก็คือในขั้นตอนของการคอมพายล์และลิงค์โปรแกรม

โดยโปรแกรมคอมพิวเตอร์ที่สร้างบน Borland C++ Builder จะใช้เวลาในการคอมพายล์และลิงค์นานกว่าโปรแกรมเดียวกันที่สร้างด้วย Borland Delphi สาเหตุเนื่องจากเวลาที่คอมพายล์งาน Borland C++ Builder จะต้องคอมพายล์องค์ประกอบของโปรแกรมที่เราสร้างขึ้น ในส่วนที่เป็นภาษา Pascal ด้วย โดยส่วนที่มีชุดคำสั่งเป็นภาษา Pascal ได้แก่ ชิ้นส่วนของโปรแกรม (Component) แบบต่าง ๆ ที่เรานำมาประกอบเป็นหน้าตาของโปรแกรม

รูปที่ 1 ตัวอย่างเกมที่สร้างด้วย Borland C++ Builder

ชิ้นส่วนของโปรแกรม (Component) ที่ปรากฏบนแถบต้นแบบชิ้นส่วนของโปรแกรม (Component Pallate) ของ Borland C++ Builder ยังคงมีชุดคำสั่งเป็นภาษา Pascal ทั้งหมด โปรแกรมส่วนที่เรียกชิ้นส่วนเหล่านี้มาใช้และชุดคำสั่งที่สร้างขึ้นบน Borland C++ Builder เท่านั้นที่เป็นชุดคำสั่งภาษา C++ (ดูรูปที่ 1)

ใช้ทรัพยากรร่วมกับ Delphi ได้

เนื่องจากต้นฉบับของ Borland C++ Builder ได้มาจาก Borland Delphi ทั้งชุด และภายในตัวโปรแกรม Borland C++ Builder เองก็มีคอมพายเลอร์ภาษา Pascal อยู่ในตัวด้วย ดังนั้นจึงทำให้เราสามารถนำทรัพยากรที่เราสร้างไว้บน Borland Delphi เช่นหน้าจอของโปรแกรมที่สร้างบน Delphi หรือชิ้นส่วนของโปรแกรม (Component) ที่เราสร้างไว้ใช้บน Delphi มาใช้ร่วมกับโปรแกรมสร้างบน Borland C++ Builder ได้แทบจะทั้งหมด

อย่างการนำหน้าจอของโปรแกรมที่เราเคยสร้างไว้บน Delphi มารวมกับโปรแกรมที่สร้างบน Borland C++ Builder จะสามารถทำได้โดยใช้เม้าส์คลิ๊กที่ปุ่มคำสั่ง Add file to project บนแถบเครื่องมือของ Borland C++ Builder ซึ่งเมื่อเสร็จขั้นตอนต่าง ๆ แล้ว Borland C++ Builder จะทำการเจนเนอเรจชุดคำสั่งที่ใช้ติดต่อระหว่าง C++ กับ Pascal สำหรับงานดังกล่าวมาให้อัตโนมัติ

แต่ทั้งนี้การนำทรัพยากรจาก Delphi มาใช้บน Borland C++ Builder จะมีบางจุดที่จะต้องปรับปรุงเล็กน้อย เช่น ในภาษา C/C++ ไม่มีชนิดของ Object ที่เป็น Real ซึ่งเราต้องเปลี่ยนให้ตรงกับภาษา C/C++ ก่อน เป็นต้น

รูปที่ 2 โปรแกรมที่จะทดลองสร้างด้วย Borland C++ Builder

Compatable กับ Borland C++ เวอร์ชัน 5.xxx

ส่วนที่เป็นตัวคอมพายเลอร์ของภาษา C/C++ ที่ใช้ใน Borland C++ Builder นั้น เป็นคอมพายเลอร์ชุดเดียวกับที่ใช้ใน Borland C++ เวอร์ชัน 5 ขึ้นไป ดังนั้นชุดคำสั่งของโปรแกรมที่สร้างด้วย Borland C++ Builder จึงสามารถนำไปคอมพายล์ด้วย Borland C++ เวอร์ชัน 5 ได้ทันที (ไม่รวมส่วนที่เป็น Object Pascal ซึ่งต้องคอมพายล์เป็น Object File ก่อนแล้วจึงนำมารวมกันในขั้นตอนของการ Link โปรแกรม) และในขณะเดียวกันโปรแกรมที่เราสร้างด้วย Borland C++ เวอร์ชัน 5 ก็จะนำมาคอมพายล์บน Borland C++ builder ได้ทันทีเช่นเดียวกัน

สนับสนุน Unicode

Borland C++ ตั้งแต่เวอร์ชัน 5 ขึ้นไป เริ่มมีการสร้างชุดคำสั่งที่สนับสนุนการใช้ตารางตัวอักษรแบบ Unicode บ้างแล้ว ใน Borland C++ Builder ก็มีชุดคำสั่งที่ใช้กับ Unicode เช่นเดียวกัน เช่นคำสั่ง swscanf () ซึ่งเป็นคำสั่งที่ใช้รับค่าข้อมูลที่ใช้รหัส Unicode เป็นต้น

แต่เนื่องจาก Unicode ยังไม่มีการนำมาใช้อย่างแพร่หลาย ทำให้ยังไม่มีการกำหนดเป็นมาตรฐานที่แน่นอน ชุดคำสั่งที่เกี่ยวข้องกับ Unicode ของ C++ ที่มีให้ใช้บน Borland C++ Builder จึงยังไม่ใช่คำสั่งมาตรฐาน แต่โดยทั่วไปแล้วคำสั่งเหล่านี้ เป็นคำสั่งที่เพิ่มขยายมาจากชุดคำสั่งที่จัดการกับตารางตัวอักษรแบบ ASCII เดิมนั่นเอง

ทดลองสร้างโปรแกรม

ในรูปที่ 2 เป็นตัวอย่างโปรแกรมที่จะทดลองสร้างบน Borland C++ Builder โดยที่บนหน้าจอของโปรแกรมชุดนี้ จะประกอบด้วยตัวฟอร์ม (Form) ซึ่งเป็นชิ้นส่วนหลักของงาน และชิ้นส่วนของโปรแกรมแบบอื่น ๆ ได้แก่ ชิ้นส่วนแบบ Panel ที่ใช้เป็นกรอบรอบ ๆ ข้อความ,ชิ้นส่วนแบบ Label ที่ใช้แสดงชื่อของงานบนตัวฟอร์ม และชิ้นส่วนแบบปุ่มคำสั่งที่ใช้สร้างปุ่มคำสั่ง "OK" ตามลำดับ

การนำชิ้นส่วนของโปรแกรมจากต้นแบบมาวางบนเฟรมที่ใช้ออกแบบหน้าจอของโปรแกรม จะทำได้โดยใช้เม้าส์คลิ๊กที่ปุ่มคำสั่งประจำชิ้นส่วนจากแถบต้นแบบของชิ้นส่วน (Component Pallate) แล้วมาคลิ๊กบนจุดที่เราจะวางชิ้นส่วนของโปรแกรมชุดนั้น ๆ บนเฟรมที่ใช้ออกแบบหน้าจอ เราจะได้ชิ้นส่วนที่ต้องการทันที

การปรับแต่งองค์ประกอบของโปรแกรม

ฟอร์ม (Form)

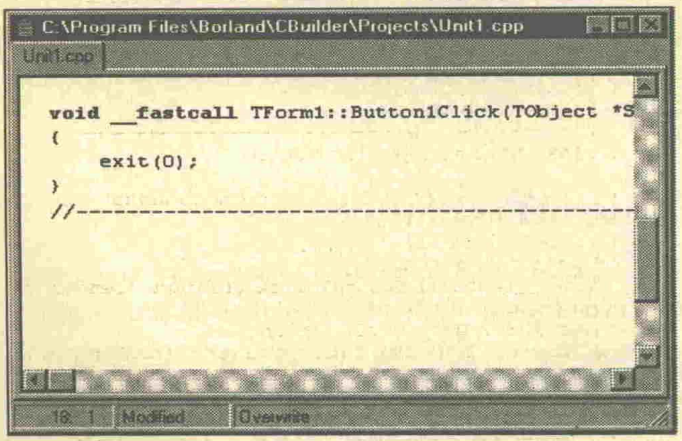
สำหรับการกำหนดคุณสมบัติทางกายภาพ (Properties) ของตัวฟอร์ม ซึ่งเป็นชิ้นส่วนหลักของโปรแกรมชุดนี้ เราจะทำการเปลี่ยนข้อความที่อยู่ในคุณสมบัติหัวข้อ "Caption" เป็น "ทดสอบการสร้างโปรแกรม" และคุณสมบัติหัวข้อ "BorderStyle" เป็น "bsDialog" ตามลำดับ

ชิ้นส่วนของโปรแกรมที่ใช้เป็นกรอบรอบข้อความ (Panel)

ชิ้นส่วนของโปรแกรมที่ใช้เป็นเส้นกรอบรอบ ๆ ข้อความหรือ Panel จะได้จากต้นแบบชิ้นส่วนของโปรแกรมชุด Standard ของ Borland C++ Builder โดยที่ชิ้นส่วนของโปรแกรมชุดนี้เราจะปรับแต่งโดยใช้เม้าส์ขยับเส้นกรอบให้คลุมข้อความที่เราสร้างไว้เท่านั้น (ในขั้นตอนการสร้างควรจะสร้างเส้นกรอบก่อน แล้วจึงค่อยสร้างข้อความภายในเส้นกรอบภายหลัง)

ชิ้นส่วนที่ใช้แสดงข้อความ (Label)

ชิ้นส่วนของโปรแกรมที่ใช้แสดงข้อความ (Label) นี้ จะได้มาจากต้นแบบชิ้นส่วนของโปรแกรม (Component Pallate) ชุด Standard เช่นเดียวกับชิ้นส่วนของโปรแกรมแบบเส้นกรอบ (Panel) โดยเราจะเปลี่ยนคุณสมบัติในหัวข้อ "Font" ของชิ้นส่วนดังกล่าว เป็นตัวอักษรแบบ EucroseaUPC ขนาด 32 จุด ตัวหนา และกำหนดสีให้เป็นสีน้ำเงิน ตามลำดับ พร้อมกับเปลี่ยนข้อความที่อยู่ในคุณสมบัติหัวข้อ Caption เป็น "ทดลองสร้างโปรแกรมด้วย C++ Builder"



รูปที่ 3 ชุดคำสั่งที่จะแทรกในปุ่มคำสั่ง "OK"

ชิ้นส่วนที่ใช้เป็นปุ่มคำสั่ง "OK" ของโปรแกรม (Button)

ชิ้นส่วนของโปรแกรมที่เราใช้สร้างเป็นปุ่มคำสั่ง จะได้มาจากต้นแบบชิ้นส่วนของโปรแกรมชุด Standard ของ Borland C++ Builder เช่นเดียวกัน โดยที่ชิ้นส่วนรูปปุ่มคำสั่งชุดนี้ เราจะเปลี่ยนข้อความที่อยู่ในคุณสมบัติของชิ้นส่วนในหัวข้อ Caption เป็น "OK" ตามที่เห็นในตัวอย่าง

คุณสมบัติหัวข้อ Caption ของชิ้นส่วนต่าง ๆ จะเป็นคุณสมบัติที่ใช้แสดงข้อความที่ปรากฏบนตัวชิ้นส่วนนั้น ๆ การเปลี่ยนข้อความเราสามารถพิมพ์ข้อความทับข้อความเดิม ในช่อง Value ของคุณสมบัติในหัวข้อนี้บนไดอะล็อก Object Inspector ได้ทันที

การแทรกชุดคำสั่ง

เป้าหมายของการสร้างโปรแกรมชุดนี้ มีเพียงต้องการแสดงหน้าจอของโปรแกรมบนจอภาพเวลาที่โปรแกรมทำงานเท่านั้น เมื่อต้องการปิดโปรแกรม เราจะกำหนดให้ผู้ใช้ใช้เม้าส์คลิ๊กที่ปุ่มคำสั่ง OK ของโปรแกรม ซึ่งจะเป็นการออกจากโปรแกรมชุดนี้ทันที

สำหรับชุดคำสั่งที่เราจะแทรกลงไปในชิ้นส่วนต่าง ๆ ของโปรแกรมชุดนี้ จะมีเพียงแห่งเดียวคือที่ปุ่มคำสั่ง OK ของโปรแกรม โดยชุดคำสั่งที่เราจะใช้ลงในปุ่มคำสั่งชุดนี้จะมีเพียงคำสั่งเดียวคือคำสั่ง exit (0) ; ดังตัวอย่างในรูปที่ 3

ขั้นตอนการแทรกชุดคำสั่งลงในชิ้นส่วนแบบต่าง ๆ ที่เรานำมาประกอบเป็นตัวโปรแกรม จะทำโดย

1. ใช้เม้าส์คลิ๊กที่ชิ้นส่วนของโปรแกรมที่เราจะแทรกชุดคำสั่ง
2. คลิ๊กบนลิ้นชักหัวข้อ "Event" ของไดอะล็อก Object Inspector ของ Borland C++ Builder
3. ดับเบิ้ลคลิ๊กที่ช่อง Value ของชื่อเหตุการณ์ที่เราจะแทรกชุดคำสั่งเพื่อควบคุมการ ทำงานของชิ้นส่วนนั้น ๆ

เมื่อเราทำการจัดเก็บโปรแกรมที่เราสร้างลงในฮาร์ดดิสก์ Borland C++ Builder จะทำการจัดเก็บงานทั้งหมดในแฟ้มต่าง ๆ โดยมีการแบ่งเป็น 2 ส่วนดังนี้

1. ส่วนที่เป็นหน้าจอของโปรแกรม ส่วนนี้จะเรียกว่า Unit เช่นเดียวกับโปรแกรมที่สร้างด้วย Delphi โดยถ้าเราไม่ได้กำหนดชื่อเป็นอย่างอื่น ๆ Borland C++ Builder จะกำหนดชื่อเป็น unit1,unit2,... ตามจำนวนชุดของหน้าจอที่เราเตรียมไว้สำหรับในงานของเรา ส่วนที่เป็นหน้าจอของโปรแกรมจะถูกเก็บในแฟ้มต่าง ๆ ดังนี้

- unit1.cpp เป็นส่วนที่ดำเนินกิจกรรมต่าง ๆ บนส่วนหน้าจอของโปรแกรม เป็นภาษา C++

- unit1.dfm เป็นส่วนที่เก็บหน้าตาของหน้าจอที่เราสร้างคำสั่งภายในจะเป็น Pascal

- unit1.h เป็น header file ที่มีมาโครสำหรับเชื่อมโยงชุดคำสั่ง C++ กับ Pascal ที่ปรากฏอยู่ในตัวโปรแกรมที่สร้างขึ้น

สำหรับรายละเอียดของแฟ้มต่าง ๆ ในส่วนนี้จะอยู่ในลิสติงที่ 1, 2 และ 3 ตามลำดับ

2. ส่วนโปรแกรมหลักของงาน Borland C++ Builder จะเก็บในแฟ้มต่าง ๆ ดังนี้

- Project1.cpp เป็นส่วนโปรแกรมหลัก ที่เรียกส่วนหน้าจอ (Unit) ขึ้นมาทำงาน

- Project1.mak เป็น Make file ที่กำหนดเงื่อนไขการคอมพายล์ของโปรแกรมชุดนี้

- Project1.res เป็นแฟ้มเก็บข้อมูลเกี่ยวกับ Resource อื่น ๆ ที่เกี่ยวข้องกับโปรแกรมชุดนี้

รายละเอียดของ project1.cpp และ project1.mak จะอยู่ในลิสติงที่ 4 และ 5 ตามลำดับ

ตัวอย่างนี้จะใช้เวลาในการคอมพายล์ประมาณ 70 วินาทีบนเครื่อง Pentium-75 ซึ่ง Execute file ที่ได้มีขนาดไม่ใหญ่นัก

ถ้าเราดูภาพรวม ๆ ของ Borland C++ Builder รุ่นนี้แล้ว ดูจะเป็นคอมพายเลอร์ที่สร้างขึ้นเพื่อสนองความต้องการของโปรแกรมเมอร์ C/C++ โดยเฉพาะ ซึ่งกลุ่มลูกค้าของบอร์แลนด์กลุ่มนี้มีไม่น้อยทีเดียว สำหรับความรู้สึกของผมแล้ว Borland C++ Builder รุ่นนี้จะเป็นรุ่นขัดตาทัพหรือรุ่นชิมรางเพื่อให้ผู้ใช้สร้างความคุ้นเคยมากกว่าจะเป็นรุ่นที่ต้องการทำตลาดอย่างจริงจัง และเชื่อว่าทางบอร์แลนด์คงจะผลิต Borland C++ Builder รุ่นใหม่ออกมาในไม่ช้านี้อย่างแน่นอน ลองติดตามดูนะครับ

ลิสติงที่ 1 UNIT1.CPP

```
001 : //---------------------------------------------------------
002 : #include <vcl.h>
003 : #pragma hdrstop
004 :
005 : #include "Unit1.h"
006 : //---------------------------------------------------------
007 : #pragma resource "*.dfm"
008 : TForm1 *Form1;
009 : //---------------------------------------------------------
010 : __fastcall TForm1::TForm1 (TComponent* Owner)
011 : TForm (Owner)
012 :
013 :
014 : //---------------------------------------------------------
015 :
016 : void __fastcall TForm1::Button1Click
      TObject *Sender)
017 :
018 : exit (0) ;
019 :
020 : //---------------------------------------------------------
```

ลิสติงที่ 2 UNIT1.DFM

```
001 : object Form1: TForm1
002 : Left = 176
003 : Top = 109
004 : BorderStyle = bsDialog
005 : Caption = 'ทดสอบการสร้างโปรแกรม'
006 : ClientHeight = 273
007 : ClientWidth = 425
008 : Font.Charset = DEFAULT_CHARSET
009 : Font.Color = clWindowText
010 : Font.Height = -11
011 : Font.Name = 'MS Sans Serif'
012 : Font.Style = []
013 : PixelsPerInch = 96
014 : TextHeight = 14
015 : object Panel1: TPanel
016 : Left = 16
017 : Top = 16
018 : Width = 393
019 : Height = 57
020 : Caption = 'Panel1'
021 : TabOrder = 0
022 : object Label1: TLabel
023 : Left = 8
024 : Top = 9
025 : Width = 377
026 : Height = 40
027 : Alignment = taCenter
028 : Caption = 'ทดลองสร้างโปรแกรมด้วย C++ Builder'
029 : Font.Charset = THAI_CHARSET
030 : Font.Color = clBlue
031 : Font.Height = -32
032 : Font.Name = 'EucrosiaUPC'
033 : Font.Style = [fsBold]
034 : ParentFont = False
035 : end
036 : end
037 : object Button1: TButton
038 : Left = 168
039 : Top = 160
040 : Width = 75
041 : Height = 25
042 : Caption = 'OK'
043 : TabOrder = 1
044 : OnClick = Button1Click
045 : end
046 : end
```

ลิสติงที่ 3 UNIT1.H

```
001 : //---------------------------------------------------------
002 : #ifndef Unit1H
003 : #define Unit1H
004 : //---------------------------------------------------------
005 : #include <vcl.hpp>
006 : #include <vcl.hpp>
007 : #include <vcl.hpp>
008 : #include <vcl.hpp>
009 : #include <vcl.hpp>
010 : //---------------------------------------------------------
011 : class TForm1 : public TForm
012 :
013 : __published:// IDE-managed Components
014 : TPanel *Panel1;
015 : TLabel *Label1;
016 : TButton *Button1;
017 : void __fastcall Button1Click (TObject *Sender) ;
018 : private:// User declarations
019 : public:// User declarations
020 : __fastcall TForm1 (TComponent* Owner) ;
021 : ;
022 : //---------------------------------------------------------
023 : extern TForm1 *Form1;
024 : //---------------------------------------------------------
025 : #endif
```

```
001 : //---------------------------------------------
002 : #include <vcl.h>
003 : #pragma hdrstop
004 : //---------------------------------------------
005 : USEFORM ("Unit1.cpp", Form1) ;
006 : USERES ("Project1.res") ;
007 : //---------------------------------------------
008 : WINAPI WinMain (HINSTANCE, HINSTANCE, LPSTR, int)
009 :
010 : try
011 :
012 : Application->Initialize () ;
013 : Application->CreateForm (__classid (TForm1) , &Form1) ;
014 : Application->Run () ;
015 :
016 : catch  (Exception &exception)
017 :
018 : Application->ShowException (&exception) ;
019 :
020 : return 0;
021 :
022 : //---------------------------------------------
```

ลิสติงที่ 4

PROJECT1.CPP

ลิสติงที่ 5 PROJECT.MAK

```
001 : # ---------------------------------------------
002 : VERSION = BCB.01
003 : # ---------------------------------------------
004 : !ifndef BCB
005 : BCB = $ (MAKEDIR)
006 : !endif
007 : # ---------------------------------------------
008 : PROJECT = Project1.exe
009 : OBJFILES = Project1.obj Unit1.obj
010 : RESFILES = Project1.res
011 : RESDEPEN = $ (RESFILES)  Unit1.dfm
012 : LIBFILES =
013 : DEFFILE =
014 : # ---------------------------------------------
015 : CFLAG1 = -Od -Hc -w -k -r- -y -v -vi- -c -a4 -b- -w-par -w-inl -Vx -Ve -x
016 : CFLAG2 = -I$ (BCB) ;$ (BCB) -H=$ (BCB) .csm
017 : PFLAGS = -AWinTypes=Windows;WinProcs=Windows;DbiTypes=BDE;DbiProcs=BDE;DbiErrs=BDE \plain
018 : -U$ (BCB) ;$ (BCB) -I$ (BCB) ;$ (BCB) -v -$Y -$W \plain
019 : -$O- -JPHNV -M
020 : RFLAGS = -i$ (BCB) ;$ (BCB)
021 : LFLAGS = -L$ (BCB) ;$ (BCB) -aa -Tpe -x -v -V4.0
022 : IFLAGS =
023 : LINKER = ilink32
024 : # ---------------------------------------------
025 : ALLOBJ = c0w32.obj $ (OBJFILES)
026 : ALLRES = $ (RESFILES)
027 : ALLLIB = $ (LIBFILES)  vcl.lib import32.lib cp32mt.lib
028 : # ---------------------------------------------
029 : .autodepend
030 :
031 : $ (PROJECT) : $ (OBJFILES)  $ (RESDEPEN)  $ (DEFFILE)
032 : $ (BCB)  (LINKER)  @&&!
033 : $ (LFLAGS)  +
034 : $ (ALLOBJ) , +
035 : $ (PROJECT) ,, +
036 : $ (ALLLIB) , +
037 : $ (DEFFILE) , +
038 : $ (ALLRES)
039 : !
040 :
041 : .pas.hpp:
042 : $ (BCB) $ (PFLAGS)  $**
043 :
044 : .pas.obj:
045 : $ (BCB) $ (PFLAGS)  $**
046 :
047 : .cpp.obj:
048 : $ (BCB) $ (CFLAG1)  $ (CFLAG2)  -o$* $*
049 :
050 : .c.obj:
051 : $ (BCB) $ (CFLAG1)  $ (CFLAG2)  -o$* $**
052 :
053 : .rc.res:
054 : $ (BCB) $ (RFLAGS)  $<
055 : #---------------------------------------------
```